Hey, I Already Did That!

==========

Commander Lambda uses an automated algorithm to assign minions randomly to tasks, in order to keep her minions on their toes. But you've noticed a flaw in the algorithm - it eventually loops back on itself, so that instead of assigning new minions as it iterates, it gets stuck in a cycle of values so that the same minions end up doing the same tasks over and over again. You think proving this to Commander Lambda will help you make a case for your next promotion.

You have worked out that the algorithm has the following process:

1) Start with a random minion ID n, which is a nonnegative integer of length k in base b

2) Define x and y as integers of length k. x has the digits of n in descending order, and y has the digits of n in ascending order

3) Define z = x - y. Add leading zeros to z to maintain length k if necessary

4) Assign n = z to get the next minion ID, and go back to step 2

For example, given minion ID n = 1211, k = 4, b = 10, then x = 2111, y = 1112 and z = 2111 - 1112 = 0999. Then the next minion ID will be n = 0999 and the algorithm iterates again: x = 9990, y = 0999 and z = 9990 - 0999 = 8991, and so on.

Depending on the values of n, k (derived from n), and b, at some point the algorithm reaches a cycle, such as by reaching a constant value. For example, starting with n = 210022, k = 6, b = 3, the algorithm will reach the cycle of values [210111, 122221, 102212] and it will stay in this cycle no matter how many times it continues iterating. Starting with n = 1211, the routine will reach the integer 6174, and since 7641 - 1467 is 6174, it will stay as that value no matter how many times it iterates.

Given a minion ID as a string n representing a nonnegative integer of length k in base b, where 2 <= k <= 9 and 2 <= b <= 10, write a function answer(n, b) which returns the length of the ending cycle of the algorithm above starting with n. For instance, in the example above, answer(210022, 3) would return 3, since iterating on 102212 would return to 210111 when done in base 3. If the algorithm reaches a constant, such as 0, then the length is 1.

Test cases

==========

Inputs:(string) n = "1211"

(int) b = 10

Output:

(int) 1

Inputs:(string) n = "210022"

(int) b = 3

Output:

(int)3

Use verify [file] to test your solution and see how it does. When you are finished editing your code, use submit [file] to submit your answer. If your solution passes the test cases, it will be removed from your home folder.